Assuring Standard Conformance of Partial Interfaces

Hardi Hungar*

Institute of Transportation Systems
German Aerospace Center (DLR)
Braunschweig, Germany
hardi.hungar@dlr.de

Abstract: A current standardization effort for track-side equipment in German railways faces the difficulty of having to proceed incrementally. This means that only some of the interfaces of complex entities like interlocking controllers are specified while others remain under the control of the diverse manufacturers. As these other interfaces are necessary for testing the specified ones for standard conformance, a specific approach has to be devised to be able to achieve this goal. This paper presents the problem in its practical setting and the way it is intended to be solved.

1 Problem Statement

1.1 General Setting

Notwithstanding the by now more than twenty years of efforts of standardizing railway control systems in Europe, proprietary interfaces and resulting incompatibilities between equipment components are still abundant. Any attempt at improving the situation faces the difficulty that the necessity of keeping the railway system in operation—defective equipment has to be replaced—and political demands—e.g. timeframes for new lines are set by third parties—that often a compromise between systematic and pragmatic solutions has to be found. Adding to this are the high costs of buying equipment and bringing it into operation.

An approach currently employed by German Railways is to incrementally specify the interface behavior of new equipment components. I.e., only one of the interfaces of an interlocking system is specified and shall be tested: the *target interface*. The other interfaces remain to be considered in the future. Current specifications define essentially the way the systems shall exchange information. Typical exchanges consist of a few (two to five) telegrams being sent back and forth. The specifications do not address core functionality of the systems, though it is a goal to ultimately extend the specifications to that level

*This research has been funded partially by the German Federal Ministry for Economic Affairs and Energy (BMWi) as part of the project NeGSt under Grant No. 19P11001A, and by the Ministry for Science and Culture of Lower Saxony on Application No. ZW3-80134621.

of completeness.

To define only single interfaces simplifies the task of standardisation as not everything has to be done at once. But the downside of this pragmatic approach is that one faces an inherent difficulty when it comes to testing. To drive the target interface (and observe the correct interpretation of messages received over it), it is usually necessary to have access to (all the) other interfaces. Specification is easier by far—one can "internalize" the uncontrolled interfaces by subsuming everything in internal behavior of a specification automaton. In the following, the problem seting will be detailed.

1.2 The Test Object

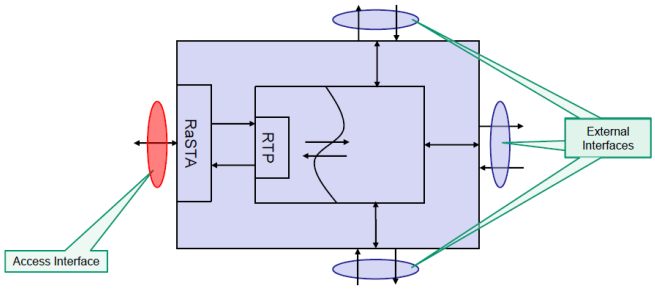

Figure 1: Schema of a system with four interfaces, of which one is to be specified.

Fig. 1 shows a schematic view of a system. The ellipses mark external interfaces to neighboring systems. The interface on the left is the target of the specification and is to be tested. The figure details relevant internals of this interface in distinguishing between its physical realisation through the *Rail Safe Transport Application* (RaSTA), and its logical view in terms of the *Rail Technical Protocol* (RTP). The functional specification to be tested addresses the RTP. The implemented systems do offer only the RaSTA interface (the figure does not depict the real architecture of the system, RaSTA need not be realised as a separate module). Therefore, testing has to bridge the gap between the protocols. Though this requires a careful translation of the specification to the RaSTA level and a few additional test cases dealing with RaSTA itself, it is not an uncommon or problematic situation.[1]

In the most interesting (and important) cases, the system is of generic nature. It consists of a collecction of software modules which have to be instantiated and parametrised to get a working device ready for deployment (and testing). The typical example for that is an interlocking controller, which contains instantiated modules for each track element in the concrete layout where it is to be deployed. The test goal is to ascertain the compliance of the target interface in each instantiation. And the usual way such systems are tested is to instantiate them for a complex track layout which covers all relevant cases of combinations

[1] The technical details of the differences between RaSTA and RTP are of course important in practice but will not be discussed in depth here.

of track elements. Manufacturers of such systems use elaborated test layouts, which have been developed over years of test practice.

A different view of testing a generic system is the following. To get meaningful behavior, the first step in testing the generic item is to instanitate it to some particular track layout. Only then the usual view of testing a reactive system, sending stimuli and observing reactions, is adopted. In practice, the generic system is instantiated only once, and then all tests are performed on one and the same instantiation. It is done in this way because instantiation is a rather difficult and partly manual step.

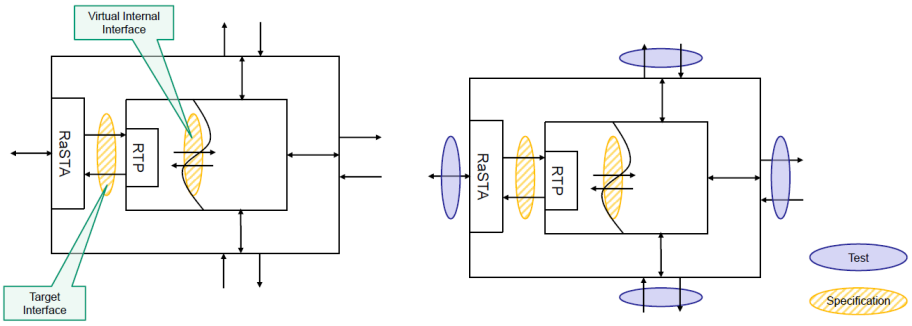

Figure 2: The specification and the test view of the system.

1.3 The Specification

More of a challenge are the ramifications of the fact that it is only the target interface which is specified (and standardized), and not the other external interfaces. To do without, the specification introduces an additional (virtual) internal interface, which replaces all the others. Formally, the specification can be viewed as a relation

$$Spec(v_{RTP}, e_{RTP}, v_{int}, e_{int})$$

between variables (v) shared on the RTP and internal interface, and events (e) on those interfaces. For its interpretation on the implemented systems, a relation between the internal interface and the external ones haa to be added:

$$\exists v_{int}, e_{int} \,.\, Spec(v_{RTP}, e_{RTP}, v_{int}, e_{int}) \;\wedge\; IntExt(v_{int}, e_{int}, v_{ext}, e_{ext})$$

Fig. 2 illustrates this situation. On the left side, the interfaces which appear in the specification ($Spec$) are shown. On the right, they are placed in the context fo the real system.

In the existing specifications, $IntExt$ is only given informally, mainly by naming variables and events with are related to functionalities of the system.

Technically, $Spec$ is given by a UML state machine, which reads and writes shared variables and exchanges events. For the interface betwen two neighboring interlocking sys-

tems, there are roughly 100 simple states in $Spec$, and a few hundred transitions. The internal interface has about 30 variables and events. and the target interface four classes of messages which are exchanged.

Genericity is handled in a rather weak way. For the interlocking interface, simply different (sub) state machines are provided for different classes of layouts of single tracks connecting two adjacent interlocking control areas. For potentially varying numbers of track elements (light signals, track occupancy detection circuitry, etc.), the generic UML model contains a number which is deemed an appropriate maximum necessary for constructing all meaningful instances. There is no formal notion of instantiation.

From a systematic point of view, this kind of formalisation must be considered as unsatisfactory. For practical purposes it might be sufficient for specifying the interfaces. The main functionality, which really depends deeply on the particular track layout, is not addressed in the interface specification. Therefore, only few and also rather simple cases have to be distinguished, and the instantiation is rather straightforward.

All of this—internalising other interfaces and informally handling genericity—works for the specification. Everything which happens on the target interface of an instantiation can be judged whether it conforms to the specification: The appropriate substate is selected, and if there is some behavior on the virtual interface, the observations is ok. Otherwise, it is not.

Testing can of course not be done in terms of the internal specification interface and by selecting substates. It needs a concrete instantiation, and the real behavior on the masked interfaces. I.e., test cases and test execution have to take the view of Fig. 1, while their derivation must refer to the right part of Fig. 2. The problem is acerbated by the unavailability of a precise relation between internal and masked interfaces ($IntExt$). In current practice, such a relation does not even exist: There are considerable differences between the masked interfaces (whose standardisation is yet to be initiated) in the different manufacturers' implementations of the devices, as already mentioned above.

Summarizing, the problem of testing the conformance of a partial interface of a system has to deal with instantiations, with the specification view and the real architecture.

2 Solution Approach

The envisaged test architecture is depicted in Fig. 3. Its components and its operation, in particular, how it copes with the problems sketched in the previous section, is explained in the following.

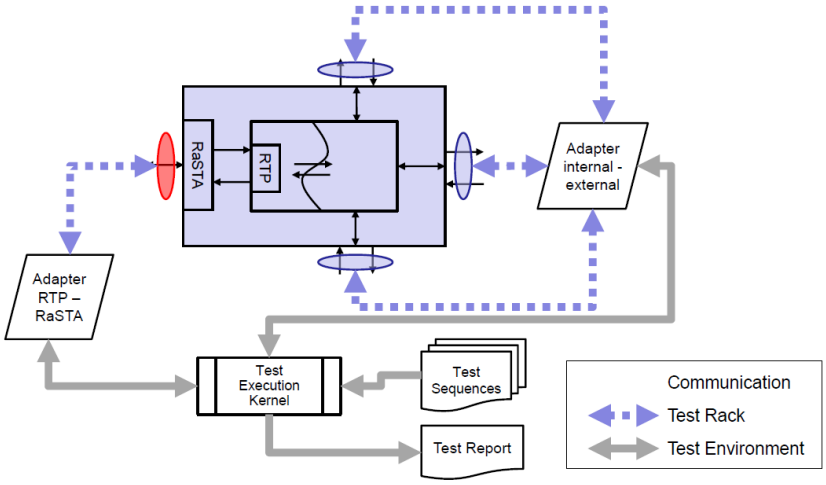

Figure 3: Components of the test architecture.

2.1 Interfaces

Differences between the implementations (i.e., $IntExt$) by different manufacturers call for integrating them into the test process in some way. Our solution relies on the assumed ability of the manufacturers of bridging the gap between (virtual) internal messages and commands and externals.

The test rack adds two components to the test object:

Adapter internal-external: The manufacturer shall provide a module which translates between internal and masked interfaces. For its realization, interface drivers, simulators, or existing test interfaces accessing internals of the device may be used. Even a test engineer performing manual steps may be integrated via a suitable interface component.

Adapter RaSTA-RTP: This module is to be provided by the test laboratory. It does not depend on specifics of the test object, as the RaSTA protocol and the representation of RTP messages within RaSTA are stadardized.

The test rack serves to provide the test object with an interface which is on the same level of abstraction as the specification.

2.2 Test Execution

Except for the interface adapters, the components of the test architecture are rather standard.

Test Execution Kernel: The kernel controls the test execution, i.e., it initializes the test objects, starts test sequences (including parameter completion in advanced scenarios), protocols the results, performs corrective actions (breaks and restarts if necessary) and generally monitors the execution. The kernel will be partially automatized.

Test Sequences: A data base with test sequences.

Test Report: A data base for detailed result data and accumulated reports.

The tests of the systems must be executed in real-time, as the systems themselves and not the software within some simulation environment will be analysed.

2.3 Genericity and Test Sequence Generation

When testing complex state-carrying communication behaviors, *test cases* covering single exchange runs are arranged into *test sequences*. A test sequence begins in an intial system state and includes a number of test cases, arranged in a way that the terminal state of each test case fits the start condition of the one following it. Auxiliary sequences establishing the start condition of test cases which would otherwise be difficult to reach may be used additionally. The set of test sequences must of course include at least one instance of each test case needed to reach the desired test coverage.

The interface standarisation activities employ such a two-stage approach in constructing the test set. First, all relevant functional exchanges on the target interface are derived form the UML state machine. For the interlocking interface, a typical test case refers to one interconnecting track. It may depend on the type of the interconnection point (e.g., with or without light signal), and also may require a certain number track elements along the way. In a second step, the set of test sequences is defined. In contrast to the generic test cases, the sequences are specifically mapped to the test track layout.

Test sequence construction is in general a complex task which has to consider several aspects. If the probability of failing test cases is not negligible, the sequences must either be short enough that not too many test cases are not executed due to previous failures, or they must be made robust enough by incorporating failure management routines. Therefore, it is important for the efficiency of the test set to be able to flexibly build the sequences.

Currently, test cases and test sequences are constructed manually, and formats as well as procedures are not formalized. In particular, there is no way to systematically explore the arrangement of test cases in sequences. For the test environment which is being developed, it is planned to define a formal relation between the set of test cases and their sequentialisation. This includes a formalisation of the conditions on track layouts on the level of

test cases. Based on this, it becomes possible to automatise the process of test sequence generation. Also the first step in the process, the derivation of test cases from the interface specification, is intended for mechanisation. Here too, part of the challenge lies in the instantiation to (constraints of) the track layout.

3 Realising the Test Environment

3.1 Plan Sketch

The test environment is intended to be realised incrementally. Starting point are available interface and test case specifications. Besides the interlocking-interlocking interface, also the communication between interlocking and radio block center[2] is already defined.

As a first step, these will be further formalised, a test track layout will be defined and some sample test sequences wil be constructed. The test excecution will be animated on a prototypical deomnstrator. A software simulation will take the place of the real test object. The main goal of this demonstration is to explore the details of formalisation of test sequences and track layouts, to identify potential problems, and to precisely specify the requirements on formats and components.

Following that, it is intended to construct a first operational version fo the test environment, capable of testing the interfaces defined so far. Its internal processes and components will in the beginning be limited in scope, capability, formalisation and automatisation. Incrementally, the concepts explained in the previous sections are going to be integrated in a form and as far as they are useful and feasable.

3.2 Validation

An imprtant aspect of the whole endeavour is the role the tests shall play and how meaningful their results will be.

To be able to make qualified assertions of standard conformance, several arguments have to be spelled out. For instance, the correctness and completeness, resp., sufficient coverage, of the test cases wrt. the specification has to be checked. This involves a careful application of techniques and methods form the domain of model based testing. The adapter design and validation will have to cope with the common problems of crossing abstraction levels (namely atomicity and timing issues as well as value concretizations). For the internal-external adapter a monitoring concept which observes dynamically its operation is envisioned. The user interface of an interlocking systems provides many informations about internal states and thus qualifies as an adequate point of observation.

[2] A device which keeps the radio communication between interlockings and trains.

4 Conclusion

An approach to solve the problem of checking the standard conformance of complex rail devices wrt. partial interface specifications has been presented. The standardization concerns the functional interface aspect of the systems, including those real-time properties which are relevant to the systems' function (and safe behavior). The goal is to assert that system passing the test will be compatible in operation. The German Aerospace Center is involved in several ongoing activities which relate to the specific topic described here.